How to write good code

Damiano Varagnolo

Department of Computer Science, Electrical & Space Engineering
Luleå University of Technology

Version 1.0.1 - December 9, 2018

”Any fool can write code that a computer can understand. Good programmers write code that humans can understand.” - Martin Fowler

1 General tips

- Don’t use accented letters or extended ASCII characters;
- write everything in plain English;
- leave no warnings behind;
- use capitalization instead of underscores;
- get rid of useless code: the less code you have, the less there is to maintain and to fix;
- hard-coding = EVIL;
- prematurely optimizing the code = ALSO EVIL. Citing Donald Knuth: “*Premature optimization is the root of all evil.*”. Simple code is faster to write, faster to understand when you return to it later, and faster to debug. **Write programs like stereotypical old grannies drive.**

2 Tips on how to give names in general

- Always give self-descriptive names. Do not try to make names short, you will lose more time trying to decipher the code;
- calling variables `a`, `b`, `c`, etc. implies that it will be impossible to search for instances of them using a simple text editor; moreover you will need to read the code to understand what they are;
- things that are related should have similar names. E.g., `SetPintleOpening` and `SetPintleClosing`;
- avoid misspellings: by misspelling some function and variable names, and spelling it correctly in others (such as `SetPintleOpening` and `SetPintalClosing`) you prevent the possibility of using effective text-search techniques;
- keep the names up-to-date: if the functionality of something changes, then update its name;
- use pronounceable names. E.g., `iGenymdhms` is bad; `iGenerationTimeStamp` is good.

3 Tips on how to give names to variables

- Use Hungarian-like notation for prefixes to indicate "what is what", i.e.,:
  - `t` for objects and structures;
  - `h` for handles/pointers;
  - `i` for integers;
  - `f` for floats;
  - `d` for doubles;
  - `str` for strings;
  - `ch` for chars;
  - `b` for booleans;
  - `a` for arrays;
  - `aa` matrices (i.e., arrays of arrays);
  - `aaa` 3D matrices (i.e., arrays of arrays of arrays), and so on.

  For example, if you read in some code `aiNumberOfCellsPerQuadrant` then you know immediately that this is a vector of integers;

- use names indicating "objects". For example, `aiNumberOfCellsPerQuadrant` indicates a "thing" and not a function. This is in line with the intuition that a "variable" represents something concrete (e.g., a value).

4 Tips on how to give names to functions

- Start the name of the function by capitalizing it. If you see `AllocateTheStorage(4)` in your code, you understand immediately that this is a function and not a variable;
- use names indicating "actions". For example, `AllocateTheStorage` indicates something that is *done* by the computer *on* something else. This is in line with the intuition that a "function" represents an action (e.g., a transformation of something into something else).

5 Tips on how to give names to classes, structures, and attributes

- Start the name of the class / structure by capitalizing it, and the name of the instances following the Hungarian notation above;
- use names reminding an "occupation". For example, `TextParser` indicates something that has a clear task to do.

6 Tips on how to write functions

- Write functions that fit on one screen;
- write error-checking code that automatically checks if there is some inconsistency;
- functions may execute more than one task. When these tasks require a few lines of code, these lines (i.e., tasks) within separate blocks of code. In other words, add some blank lines between the various tasks and put a comment at the beginning of the tasks;
- DIE, a.k.a. *Duplication Is Evil.* This is an important rule: "Every piece of knowledge must have a single, unambiguous, authoritative representation within a system." Automate repetitive tasks, i.e., if you have to do twice the same task do a function that handles that task;
- limit the line length – it takes more effort to move the eyes horizontally than vertically. That's why newspapers have very tall and narrow columns of text.

7 Tips on how to indent the code

- **BE CONSISTENT (this is a rule, not a tip)**. To strengthen the point, this is a famous haiku on the consistency of spacing:

  Anybody who mixes tabs and spaces
  for indentation
  will spend an eternity burning in hell.

- use tabs instead of spaces;
- set a tab width in the editors of 4 spaces;
- expand the code over multiple lines, and make matching parentheses appear in the same column.

8 Tips on how to write comments

- if a function implements a non-straightforward algorithm, describe it at the beginning of the function itself using a pseudo-code that mirrors your actual code;
- if the name of a function does not clearly define what it makes, put comments at the beginning of the function describing what are the inputs and the outputs;
- inside functions, put comments that split the code up into shorter tasks;
- for chunks of code that seem thorny, provide a quick explanation of what is happening.
- don't let the persons coming after you think that you are dumb. Example:

```
% Now we increase Number_of_sensors by one.
Number_of_sensors = Number_of_sensors + 1;
```

- at the same time, don't let the persons coming after you think that you are an asshole that writes purposely obfuscated code.

9 How to quantitatively measure code quality

The international standard for measuring code quality is in WTFs per minute. Try to minimize the number of WTFs you generate.

the most important tip: help those that will come after you[a]

[a]And consider that you will come after yourself.